Architecture for an efficient integration of wireless sensor networks to the Internet through Internet of Things gateways

Patrick Olivier Kamgueu, Emmanuel Nataf, Thomas Djotio

Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2017, Vol. 13(11)

DOI: 10.1177/1550147717744735
journals.sagepub.com/home/ijdsn
SAGE

Architecture for an efficient integration of wireless sensor networks to the Internet through Internet of Things gateways

Patrick Olivier Kamgueu[1,2,3], Emmanuel Nataf[2,3] and Thomas Djotio[1]

Abstract

In recent years, Internet of Things has changed the way people work and live, thereby opening new opportunities. This article describes a framework architecture for interconnecting several wireless sensor networks to the Internet to achieve the vision of Internet of Things. Connecting things together is done through gateways that act as single points of failure to bridge the connection between wireless sensor network and the traditional wired Internet. To cope with the unreliable nature of wireless links and scale to a large number of sensors and wireless sensor networks, several gateways should be installed. Given a number of previously deployed gateways, the contribution of this article is twofold. First, it focuses on choosing the most suitable ones for connecting wireless sensor networks to the Internet in a efficient and cost-effective manner, using integer linear programming to develop the mathematical model. Second, a network topology that puts the integration of wireless sensor networks to the Internet in place is built. A three-layer architecture is used in such a way that the intermediate layer adapts dynamically to network changes and evolution. Gateway selection procedure at this layer was implemented using CPLEX linear solver, and the gateway was integrated on Raspberry Pi cheaper hardware which serves as the routing protocol for low-power and lossy network root for the wireless sensor network configuration. Experiments and real deployments have been conducted to demonstrate the effectiveness of the proposed scheme.

Keywords

Internet of Things, gateway, routing protocol for low-power and lossy network, integer linear programming, CPLEX

Date received: 17 April 2017; accepted: 27 October 2017

Handling Editor: An Liu

Introduction

The Internet of Things (IoT) enables physical devices not only to be connected to the virtual world but they can also remotely receive their order from the Internet and act as access points to various Internet services.[1] In that context, *things* are entities with computing and communication capabilities, so they can be seen as devices having some level of intelligence able to interact with other devices, themselves connected to the global data network. Interconnecting devices together as well as devices to the Internet is a challenging task due to the large number of devices involved. Moreover, the heterogeneity of technologies and hardware require the definition of common languages and standards which allow them to interoperate.

[1]University of Lorraine, Nancy, France
[2]University of Yaoundé I, Yaoundé, Cameroon
[3]Centre de Recherche INRIA Nancy, Villers-lès-Nancy, France

Corresponding author:
Patrick Olivier Kamgueu, Centre de Recherche INRIA Nancy, Grand Est, 615 Rue du Jardin Botanique, 54600 Villers-lès-Nancy, France.
Email: patrick-olivier.kamgueu@inria.fr

Wireless sensor networks (WSNs) are important building blocks of IoT. The general topology of the network consists of several sensor nodes scattered in a target area where they monitor various physical phenomena. A particular node also known as *sink* is attached to a base station and it receives sensed data through relay nodes using multi-hops radio communication. Those nodes are constrained in hardware and other resources. They are low-powered, equipped with a limited storage and processing capabilities, low transmission rates over a lossy communication media, and usually operate on embedded batteries for their energy supply. Several WSNs may share the same area, each targeting a given objective or monitoring a particular phenomenon. For instance, it is possible to monitor temperature or other environmental parameters using a given network, along with the presence of detection in a particular building or smart meters polling to get information feedback to a central processing point with another network. Application scenarios of such infrastructures are numerous and encompass, but not limited to home automation and smart building to improve living comfort. In smart cities and healthcare, they offer various services to citizens and help to assist patients. When they are used in industrial process instrumentation or critical systems monitoring, they are subjected to specific delays and quality-of-service (QoS) requirements.

The aforementioned usages require a deployment of several WSNs, each with specific application goals. Moreover, WSN requirements can evolve (more sensors) and new needs that have not yet been identified may also emerge. To provide Internet connectivity for deployed WSNs, we envisaged to set up several gateways acting as a backbone framework. They bridge the Internet connection to several WSNs. A wireless network interface (IEEE 802.15.4 for example) connects the WSN while the Internet side uses a typical Ethernet network adapter or any other access technology interface such as long-term evolution (LTE, also widely known as 4G, is a standard for high-speed wireless communication for mobile devices and data terminals, based on the GSM/EDGE and UMTS technologies) or IEEE 802.11. So, they will not only enable the linking (through Internet access) of some unconnected parts of the same WSN but also provide a remote access to all sensor networks through the Internet. Due to their limited capacity (in terms of bandwidth), gateway acts as a single point of failure for Internet connection and could hinder traffic coming from the WSN side (bottleneck). Since several WSNs are set up at the same time and given the unreliability of wireless links, each connected part of the WSN should access Internet through several possible gateways. Network should adapt and change the Internet entry point depending on the availability of gateways and the amount of traffic and other network conditions (QoS requirement and application constraints).

The contribution of this article is twofold. First, given the network conditions and other constraints, it determines the most appropriate gateways from existing ones that would provide an efficient Internet connection of deployed WSNs at lower costs. Second, a three-layer architecture that dynamically adapts to network conditions is designed and the end-to-end connection between WSNs and Internet ensured.

There are two somewhat conflicting objectives which should be met:

- WSNs will seek to use the maximum available gateways to meet QoS, bandwidth, and fault tolerance. Indeed, the more the available and open gateways, the higher the bandwidth to convey data.
- Only an appropriate number of gateways should be used. These are those that match actual application requirements given current network conditions.

The main idea is to install the maximum number of gateways in the network, but to select and use a limited number of these devices. The remaining capacity (unused gateways) will overcome any future failure (gateway, and node and link breakdown) or serve for scaling purposes (addition of new nodes to the network or deploying new WSNs and services). At the same time, we aim to balance traffic load among all selected gateways, thus, the use of active resources is optimized.

The rest of this article is organized as follows. In section "Related works," related works are reviewed, followed by a description of the network model and problem statement. A formal definition of the algorithm and resolution approach is done with illustrative examples. The "System architecture" and "Routing scheme" sections highlight the system and protocol design. Then, implementation considerations and some use cases are provided. The article ends with the "Conclusion" section.

Related works

The problem of using gateways to connect WSN to the Internet to achieve the vision on IoT has already been investigated in the literature. Zachariah et al.[2] put the emphasis on the use of Bluetooth Low Energy (BLE) to establish the connection between various sensor nodes and the Internet. They advocate the use of smartphones for this purpose. Our work aims to connect WSNs to Internet or establishing communication between several unconnected WSNs. In this case, network access technologies around IEEE 802.15.4 standards and the IPv6 routing protocol for low-power and lossy network

(RPL) are more appropriate. However, providing gateways with more interfaces, including BLE, allows the support of a large variety of devices and new IoT appliances.

To design an access scheme for sensors through IoT gateways, Wang et al.[3] proposed a three-layer approach: sensor networks (*perception layer*), gateways (*network layer*), and middleware (*application layer*). They suggest the use of Lightweight Directory Access Protocol (LDAP is an open industry standard application protocol for accessing and maintaining distributed directory information services over IP networks) to name network components and resources, and thus make the naming and membership to be grouped in one of the layers. Our scheme uses the IPv6 protocol to organize sensors and gateways into subnets. The link with applications is also ensured, since IPv6 is widespread among Internet service providers. On this point, our proposal corroborates those introduced by Bimschas et al.,[4] where the middleware that facilitates communication between applications and WSN takes place on gateways.

Number and locations of gateways within WSN are crucial as they impact the network performance and lifetime. Some heuristics used for node placement or load balancing can be found in the literature.[5] Among others, we can mention recursive and iterative dominating set, augmenting and greedy placement. But these algorithms relate to wireless mesh networks (WMNs) and are not directly applicable to WSNs, especially when network topology is built as a directed acyclic graph (DAG) such as in the case of RPL.

The use of linear programming (LP) for solving gateway deployment problem is discussed by Wu and colleagues.[6,7] Once again, the proposed solutions relate to WMN and primarily aim at maximizing throughput of gateways while balancing the load between them. Boubrima et al.[8] also suggested the use of LP for gateway deployment in order to monitor air quality. Their integer programming model expresses constraints in a way that pollutant concentrations are accounted when placing sensors with the aim of achieving a bounded error at locations where no sensor is deployed while minimizing the financial cost. Our model also uses LP to express desired network characteristics as constraints, particularly in an RPL-like tree-based topology. Moreover, commands issued to open or close gateways are evaluated dynamically depending on network conditions, rather than estimated in advance for a fixed topology.

Our work is a preliminary step from those carried out by Petrolo et al.[9] Here, gateways are designed to couple semantic web technologies and cloud computing to enable various IoT applications and platform integration. Their main goal is to achieve the Cloud of Things (CoT) vision. As a result, used gateways handle semantic-like things and act as an end-point for the dynamic presentation of real-world data to consumer applications and users. They are thus used to enable a lightweight and dense deployment of services, thanks to virtualized software and technologies. This article aims to provide an architecture that ensures a flexible and robust communication between WSN and backbone Internet with low-cost hardware. In this way, our scheme takes place at routing level (network layer), whereas the above work relies on an application layer by enabling service integration and data presentation through container-based virtualization in gateways.

Design considerations

This section describes the network architecture and presents some major design considerations. The gateway selection procedure is formulated as an integer linear program (ILP) problem under several constraints. Figure 1 shows the envisioned network architecture. Gateways depicted as routers are more robust nodes than those scattered in the target area. They are constantly powered and have sufficient memory and processing resources. In the proposed model, the third version of Raspberry Pi platform[10] is used, thanks to its good hardware features and low acquisition cost. So, it is easier (in terms of investment cost) to deploy a large number of such devices in a given area of interest. Sensor nodes acquire and relay data through multihops wireless links to the border router performed by the gateway. These sensor nodes belong to separate networks depicted in Figure 1 by color codes (red, blue, or gray) depending on their application or network goal.

Network model

The target framework which is illustrated in Figure 1 network can be modeled as directed graph $G = (V, E)$, where E is the set of edges, depicting wireless reachability between neighboring nodes, and $V = V_g \cup V_n$ is the set of vertices. The latter is either a component of $V_g = \{g_1, g_2, \ldots, g_m\}$ of m gateways or component of V_n, the set of sensor nodes, which themselves belong to separate WSN scattered in the monitored field. More formally, $V_n = \bigcup_{j=1}^{k} V_{n_j}$, where k is the number of deployed WSNs. Note that the set of sensor nodes belong to separate WSNs, V_{n_j} forming a partition of V_n (since WSNs are distinct through their goal or application).

For illustration, consider the network depicted by Figure 1. Values are $k = 3$, $m = 5$, and $|V_n| = 37$. Let V_{n_1}, V_{n_2}, and V_{n_3} being WSNs, respectively, denoted by

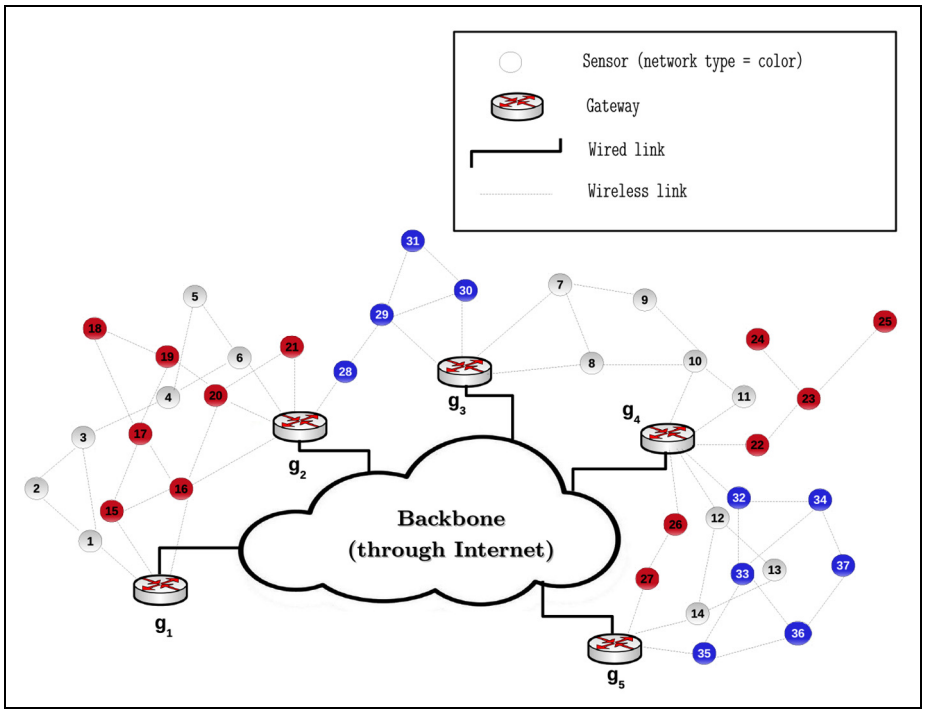


Figure 1. Backbone architecture for WSN interconnection.

color codes gray, red, and blue with the set cardinality $|V_{n_1}| = 14$, $|V_{n_2}| = 13$, and $|V_{n_3}| = 10$.

All network nodes (gateways and sensors) are assumed to be stationary once deployed. Every gateway g_i $_{i\in[1,m]}$ has a limited capacity $C(g_i) = \alpha_i$ known at the setting up step. This indicates how much traffic the gateway can handle given its bandwidth to access the Internet. For simplicity, all nodes are assumed to produce the same amount of data, thus, each gateway capacity is seen as "the number of sensors for which the latter can relay data to the Internet."

Problem formulation

To set up the network, the more the gateways, the better the overall system capacity. It is desirable to minimize their setup cost and usage. However, distributing traffic load among all open gateways is sought, as well as ensuring sufficient network resources (i.e. bandwidth capacity) to allow a *good QoS* Internet access for deployed WSNs. The main goal will be then to look for the optimal number of gateways to use, as well as their placement in the network (among all already installed gateways). More formally, the problem can be formulated as follows.

Given a network modeled by the directed graph $G = (V_g \cup V_n, E)$, where $|V_g| = m$ is the number of installed gateways, each having a capacity $C(g_i) = \alpha_i, i \in [\![1, m]\!]$ and V_n is the set of deployed sensors. Let k be WSNs each including $|V_{n_j}|$ sensor nodes, $j \in [\![1, k]\!]$. We would like to select $s \leq m$ among the exiting m gateways so that all WSN demands are met while spreading incoming traffic load as fairly as possible among the selected gateways.

When looking for solutions to the above problem, several constraints are taken into consideration:

- *WSN coverage*. Each sensor node must have an Internet access. This is possible through any gateway using multi-hops wireless communications. Indeed, there must be at least one path that connects any sensor node to at least one gateway.
- *Gateway capacity*. The number of nodes served by a gateway should not exceed its capacity. In case of more nodes supported by a given gateway than its capacity, more gateways must be added to the network (or enabled if previously deployed).
- *Gateway utilization*. A given gateway can only be used as Internet access point by the sensor nodes if at least one sensor uses the latter for its Internet access. As a result, it must be opened by the algorithm of selection.
- *Network deep*. Due to unreliability of data transmission when using wireless medium, the packet loss rate increases with distance to the base station. To increase the throughput, an additional constraint limiting the length of paths during the routing topology building step is introduced,

namely, the *maximum hop-count* noted MAX_{Deep}. Hence, a given node only uses a gateway if the latter is located at most MAX_{Deep} hops from it.

Under the aforementioned constraints, two optimization objectives are targeted:

1. *Minimize the number of opened gateways*. Some sensor nodes may be in the service area of many gateways. The selected gateway(s) must allow connection of those at the best service costs by optimizing the routing metric (e.g. hop-count, delay, or reliability).
2. *Balance the network load among the used gateways*. In other words, this relates to maximizing the use of opened gateways.

Gateway selection procedure

Computational complexity

Proposition 1: NP-hardness of gateway selection problem. The gateway selection problem as stated in "Problem formulation" section is NP-hard.

Proof. The NP-hardness of gateway selection problem is established through the reduction of the well-known capacitated facility location problem (CFLP) to it.[11]

Indeed, there is a set F of facilities able to produce each, some amount of a given product. Facilities are either open or closed and must satisfy some demands d_j from various clients $j \in D$ located in different cities. Opening a facility $i \in F$ involves an operational cost f_i and a production capacity α_i that matches the maximum demand that facility i can serve. The cost of one-unit demand from client j by the facility i is c_{ij}. The CFLP is to determine a subset $F' \subseteq F$ of facilities to open so that all client demands are met and the total operational cost related to the opening of facilities and serving the clients is minimized.

Reducing CFLP to gateway selection problem is quite simple. One can see the CFLP as an instance of this problem, where facilities match the gateways and they have the same operational cost $f_i = 1$. The service cost c_{ij} of one client j (i.e. a sensor node) by the gateway i is related to the metric weight associated to the path from a node to the selected gateway. For simplicity, the weight of paths is chosen as the hop-count. All sensor nodes are supposed to express the same demand $d_j = 1, \forall j \in D$ corresponding to the generated traffic. However, it is easy to generalize sensor demand to any arbitrary integer value. So, facility capacity is set on the amount of data that the associated gateway can relay. The value noted, α_i, is the maximum number of sensor nodes (or involved traffic) supported by the ith gateway. CFLP is then reduced to the gateway selection problem. Since the NP-hardness of CFLP is well established,[12] it can be inferred that the gateway selection problem is also NP-hard. □

Solving method

The problem is formulated as an integer linear problem (ILP). All notations are those introduced in the "Network model" section. Gateways, indexed by i, are members of V_g set, while sensors, indexed by j, are members of V_n set. Two new multidimensional variables X and Y are also introduced, where their binary values X_{ij} and Y_i are defined as follows

$$X_{ij} = \begin{cases} 1 & \text{If gateway } i \text{ serves node } j \\ 0 & \text{otherwise} \end{cases}$$
$$Y_i = \begin{cases} 1 & \text{If gateway } i \text{ is open} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Gateway selection problem can be rewritten as in the above ILP

$$\text{minimize} \left[\sum_{i=1}^{|V_g|} \sum_{j=1}^{|V_n|} c_{ij} X_{ij} \quad + \quad \sum_{i=1}^{|V_g|} Y_i \quad + \quad \underset{i \in V_g}{\sigma} \left(\sum_{j=1}^{|V_n|} X_{ij} \right) \right] \quad (2)$$

$$\text{Subject to :} \begin{cases} \sum_{i=1}^{|V_g|} X_{ij} = 1, \ \forall j \in V_n & (3) \\ \sum_{j=1}^{|V_n|} X_{ij} \leq \alpha_i, \ \forall i \in V_g & (4) \\ X_{ij} \leq Y_i, \ \forall i \in V_g, \ \forall j \in V_n & (5) \\ \sum_{i=1}^{|V_g|} c_{ij} X_{ij} \leq \mathrm{MAX}_{Deep}, \ \forall j \in V_n & (6) \\ 0 \leq X_{ij}, Y_i \leq 1, \ \forall i \in V_g, \ \forall j \in V_n & (7) \end{cases}$$

The objective function (OF) given by equation (2) seeks to minimize the number of gateways to open $\left(\sum_{i=1}^{|V_g|} Y_i \right)$, and it also minimizes the weight associated with the path (i.e. hop-count) to reach the selected gateway $\left(\sum_{i=1}^{|V_g|} \sum_{j=1}^{|V_n|} c_{ij} X_{ij} \right)$. Moreover, to consider load balancing among open gateways, a third term $\underset{i \in V_g}{\sigma} \left(\sum_{j=1}^{|V_n|} X_{ij} \right)$, is introduced as the standard deviation of opened gateways load. The smaller the value, the better the load distribution compared to the average.

The first constraint, also known as WSN coverage constraint, is described by relation (3). It ensures that each node can reach at least one gateway. Moreover, as the sum value is set to 1, this requires that every node accesses the Internet through one and only one

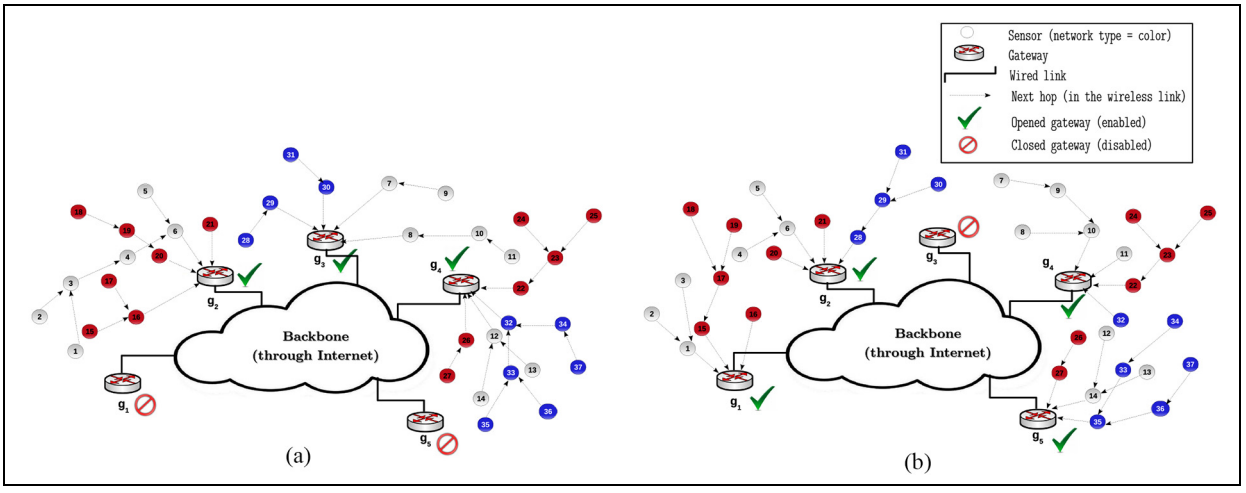


Figure 2. Routing tree associated with various input data: (a) $\alpha_i = 15$, $\forall_{i\in} V_p$ and (b) $\alpha_i = 10$, $\forall_{i\in} V_p$

gateway. Thereafter, separate routing trees can be built. The inequality (4) is the capacity constraint. It ensures that clients' demands do not exceed the selected gateway capacity. Relation (5) denotes that only opened gateways can be used as sinks for the routing. The following relation, inequality (6), restricts the coverage area of gateways through multi-hops wireless links to MAX_{Deep}. Finally, both inequalities referenced by equation (7) use binary values (0 or 1) for the ILP output variables, X and Y.

There are several heuristics[11,12] to solve the proposed ILP. We opted to use the IBM ILOG CPLEX v 12.6.3 libraries.[13] In the latter, the CPLEX optimizer looks for exact solutions for the ILP and restricts the search space using the *branch and cut* method. The model has been coded in Java programming language.

Illustrative scenarios

Three variables are given as inputs, MAX_{Deep} the maximum length of any path to a gateway, α the one-dimensional array of gateway capacities and the two-dimensional array C of c_{ij} values that provide the access cost of node j to gateway i. Two output array values are produced. The first output is a one-dimensional array Y, where $Y_i = 1$ denotes that the ith gateway is open and the value is otherwise 0. The second output is a two-dimensional array X, where $X_{ij} = 1$ denotes that sensor node j uses the ith gateway as its border router to access the Internet, otherwise the value is 0.

The network topology depicted by Figure 1 is used to illustrate how the model operates. The two-dimensional array C (5 rows × 37 columns) associated with value c_{ij}, which denotes the hop-count used as the metric and optimized by the routing, is given as follows. The infinity (depicted by dots for the sake of space) at c_{ij} indicates unreachability (through multi-hops wireless link) between gateway g_i and node j. In practice, the value was instantiated with a sufficiently high number compared to the cost of any other path

$$C = \begin{bmatrix}
1 & 2 & 2 & 3 & 4 & 4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 2 & 3 & 3 & 2 & 3 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
4 & 4 & 3 & 2 & 2 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & 1 & 2 & 3 & 2 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 2 & 3 & 3 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 2 & 2 & 3 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & 1 & 1 & 2 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 3 & 2 & 2 & 1 & 1 & 1 & 2 & 2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 2 & 3 & 3 & 1 & 2 & \cdot & \cdot & \cdot & 1 & 2 & 2 & 3 & 3 & 3 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & 2 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & 1 & \cdot & \cdot & \cdot & 3 & 2 & 3 & 1 & 2 & 3
\end{bmatrix}$$

First scenario. First, all gateway capacities are set to the same initial value 15. So, $\alpha = [15, 15, 15, 15, 15]$. The maximal path to any gateway has been restricted to $MAX_{Deep} = 4$. After instantiation and resolution, the proposed model produces as outputs the arrays X and Y as follows

$$Y = [0, 1, 1, 1, 0]$$

$$X = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Hence, based on the inputs provided, the system advocates the opening of gateways $g_2, g_3,$ and g_4, and the other should be closed. Figure 2(a) illustrates the resulting network topology. The first gateway (g_2) should be opened for WSN instances governed by gray and red color, the second (g_3) opened for gray and blue instances and the last gateway should be opened for all the three instances.

It is worth noting that load distribution (note β) related to gateways $g_2, g_3,$ and g_4 are, respectively, $\beta_2 = 13, \beta_3 = 9,$ and $\beta_4 = 15$. This represents a standard deviation of $\sigma = 2.49$. Gateway g_4 reaches its maximum capacity, while g_3 captures all nodes in its coverage area in order to provide a standard deviation that is the lowest possible for this scenario.

Second scenario. Let us consider the same inputs as the previous scenario, except gateway capacities that are now set on the same value 10, that is, $\alpha = [10, 10, 10, 10, 10]$. The model outputs the arrays below. The resulting network topology depicted by Figure 2(b) is completely different

$$Y = [1, 1, 0, 1, 1]$$

$$X = \begin{bmatrix} 1&1&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&1&1&1 \end{bmatrix}$$

Note that by setting gateway capacities to a lower value, the system behaves as expected. It suggests the opening of more entry points toward the Internet backbone (four instead of three as previously mentioned). In addition, the new distributions of gateway loads are $\beta_1 = 8$, $\beta_2 = 9$, $\beta_4 = 10$, and $\beta_5 = 10$. The resulting standard deviation is $\sigma = 0.83$, denoting better load balancing. Table 1 summarizes the system input and output data for the highlighted scenarios.

System architecture

To instantiate the described model, it is necessary to gather information about the network topology. In particular, those related to various deployed WSNs and their interconnection graph from which the optimal paths to reach gateways and the associated costs are inferred. Hence, this network map also allows the building of the two-dimensional array C used as the ILP input.

Figure 3 shows the overall functional architecture of the system. As illustrated, three main entities are required for system implementation: sensor nodes, gateways, and the centralized management station. The sequence diagram at the bottom part of the figure outlines the interactions between these entities:

1. *Vicinity information gathering*. Regularly, for every network instance, sensor nodes collect neighborhood information and forward them using multicast to base stations. Both, the base station and gateway are ensured by the same physical device. To gather vicinity data and manage topological information, a dedicated protocol (Dynamic RPL Configuration Protocol (DRPLCP))—described later under the section "Network operations"—has been designed and implemented.
2. *Shortest path graph*. Each gateway builds from the received data (neighborhood graph) the shortest path tree. The gateway plays the role of root (also the border router) while sensors are either internal nodes or leaves of the tree. The built topology is also the optimal tree for the selected routing metric (hop-count is used in the proposed model), sent by gateways to the management station. The latter aggregates that information and provides C cost matrix to the ILP engine. A given row in the matrix describes the shortest path from all nodes to the corresponding gateway.
3. *Model instantiation*. Once all model inputs are known (cost matrix, vector of capacities, and the maximum allowed length of paths), the ILP is carried out by the CPLEX solver on the management station. The gateway status (open/closed) and the corresponding roles (namely, RPL instances to configure) are then provided.
4. *Commands/order transmission*. Status and roles of gateways computed in the previous step are conveyed to the latter via the network. Gateways accommodate their behavior accordingly and update their capacity. The new gateway capacity is updated by deducting the received load from the previous capacity. More formally, $\alpha_i(t) = \alpha_i(t-1) - \beta_i(t)$, where $\alpha_i(t)$ is the gateway i computed capacity at time t, $\alpha_i(t-1)$ is the previous capacity, and $\beta_i(t)$ is the additional received load resulting from the network reconfiguration.
5. *Instance/routing configuration*. Newly opened gateways start sending parameter messages (DAG Information Objects (DIO)) for supported instances. Afterwards, sensor nodes join the network according to RPL OF, and topology is built by optimizing a given routing metric

Table 1. Summary of scenario parameters.

Scenarios	Model inputs		System outputs	
	Common parameters	Differentiated parameters	Open gateways	Traffic load distribution
First scenario	The *C* array two-dimensional (nodes' reachability to a particular gateway in terms of number of hop-counts—cf. Figure 1)	First capacitated vector $\alpha^{(1)} = [15, 15, 15, 15, 15]$	Three gateways opened: g_2, g_3, and g_4	$\sigma^{(1)} = 2.49$
Second scenario	MAX_{Deep} set to 4 (the maximum allowable length for routing paths between any node and a given gateway is 4)	Second capacitated vector $\alpha^{(2)} = [10, 10, 10, 10, 10]$	Four gateways opened: g_1, g_2, g_4, and g_5	$\sigma^{(2)} = 0.83$

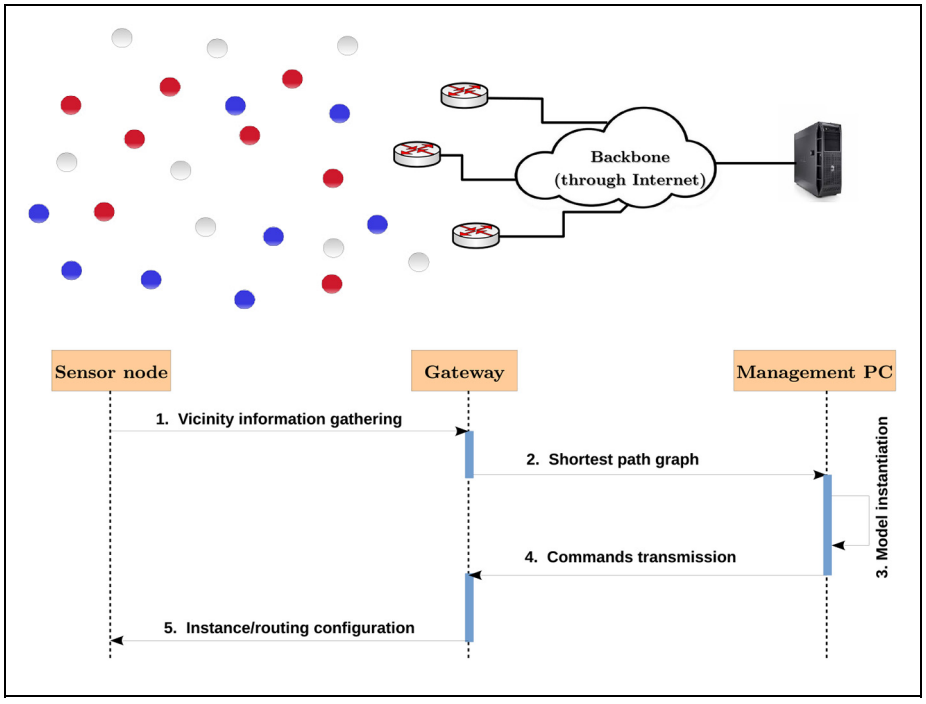


Figure 3. System architecture.

(hop-count, energy, or reliability) or a combination of them (for QoS consideration).

To account for the dynamic nature of the network, sensors keep monitoring their vicinity on a regular basis. Vicinity information is sent back to the base station only when topology changes (newly discovered node or when a neighboring node stops functioning) to reduce consumption of resources.

Routing scheme

Nodes in the WSN build the network topology according to RPL,[14] a distance vector routing protocol standardized for low-power and lossy network where multipoint-to-point is the dominant traffic pattern. Topology is organized as one or more DAG with each rooted at a single point that acts as a sink for other nodes. The setting up step starts at the sink that spreads special configuration messages in its vicinity. Those which are also called DIOs convey all configuration parameters. Later, they are forwarded hop-by-hop to flood the whole network.

Disseminated parameters include (but not limited to) instance ID, root ID, Objective Code Point (OCP), metric, and timers value. RPL instance is used to define the optimization criteria when forming a routing path according to the application goal. So, it is possible to

concurrently run several RPL instances in the network that operate independently with one another. While an instance may include several DAGs, the latter belongs to one and only one instance. The optimization goal is achieved through the OF identified by an OCP value which is the same for a given instance. OF defines a framework where routing metrics are converted to a scalar value: the *rank* is used to select node's best parent (also known as next hop) in the path toward the sink.

As illustrated, a WSN can include a two-instance RPL network where the first builds a DAG optimized to minimize latency to reach a single centralized lighting controller in a home automation application, while another network builds a DAG to maximize reliability when sending environmental data to a central server.

Up to now, Internet Engineering Task Force (IETF is a open standards organization that develops and promotes Internet standards) has standardized two OFs. The first, namely, OF0,[15] uses the hop-count as the unique parent section criterion. The second[16] is tailored for Expected Transmission Count (ETX is a link quality metric that estimates the expected number of MAC transmission (including retransmission) required to send packet over a wireless link), a metric that accounts for packet reliability. Many other OFs have been proposed by the research community. Among others, Kamgueu et al.[17] used node's residual energy as the unique routing metric and proposed a method to combine several criteria to take QoS into consideration.[18]

Implementation and deployment

System hardware

As stated earlier, the system consists of two main parts: The WSN and the Internet (through backbone infrastructure). In the proposed implementation, nodes deployed in the WSN side are TelosB type CM5000 equipped with a MSP430 microcontroller (as the CPU) and a CC2420 radio-frequency chip that uses IEEE802.15.4 wireless communication standard. Various sensor interfaces are integrated to acquire environmental measurements (i.e. temperature, humidity, and light). The TelosB was chosen, although it costs higher (about twice) than the Raspberry Pi. This is justified by many available sensors and device robustness, and besides, it is a prototype designed for the purpose of research. Depending on application requirements, cheaper nodes can be used. It is worth noting that we aim to reduce the intermediate layer hardware and setup costs, and that the WSN layer is assumed to be already deployed.

In the backbone side, the Raspberry Pi 3 is used and acts as both, the gateway and the RPL DODAG root. Indeed, the third version of this platform is a real computer that includes a quad-core ARM Cortex-A53

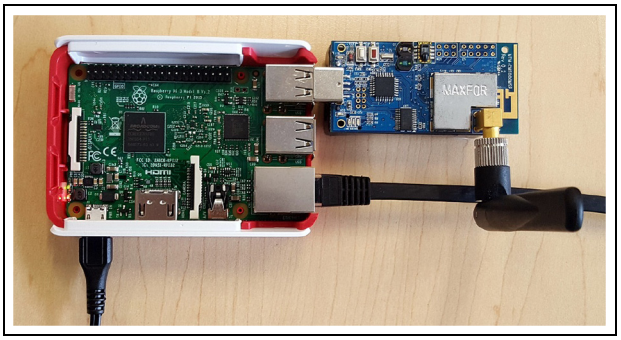


Figure 4. Internet gateway: Raspberry Pi 3 and TelosB association.

CPU clocked at 1.2 GHz with 64-bit data path, 1 GB memory, 1 Ethernet, and IEEE802.11n wireless local area network (LAN) interfaces. Moreover, it incorporates Bluetooth 4.1 and the new BLE convenient for IoT usages, in addition to 4 USB, HDMI ports, and more than 40 general-purpose input/output (GPIO) pins. The main reason of this choice is the small size of the device (the scale of a credit card), its powerfulness compared to its low cost and the ability to fully run the gateway interface program without size and other resource restrictions. Figure 4 highlights the gateway structure carried out by the association of Raspberry Pi 3 (left side) and TelosB mote (right side). The latter acts as IEEE802.15.4 interface for connecting the WSN.

Network operations

WSN operations. On the WSN field, every node runs a specific client application written in the node's flash memory prior to its deployment. A given application is recorded through a *regular* RPL instance number at the gateway. Once deployed in the network, the node begins by registering in a multicast group (using a dedicated (reserved) RPL instance) and waits until it receives a *regular* instance assigned by the RPL root. A specific protocol, namely, DRPLCP, responsible for managing the node registration and instance dispatched by the root, is designed for this purpose. The dynamic nature of node deployment and application assignment allows to set up WSNs gradually.

As soon as the node obtains a *regular* instance, it connects to the corresponding RPL routing tree and starts running the application associated with the received instance. During normal operation, nodes regularly collect information from their vicinity and forward them to their gateway. Later, this information is used by the management PC to infer the network graph and instantiate the model engine.

Gateway operations. The gateway is set up by the hardware combination, Raspberry Pi 3 and TelosB

Table 2. Simulation parameters.

Settings	Values
Application layer	Periodic sensor data collection
Transport layer	UDP
Network layer	μIPv6 + 6LoWPAN + RPL (routing)
MAC layer	Non-persistent CSMA
Radio duty cycle	ContikiMAC
PHY + radio chip	IEEE 802.15.4 with CC2420
Wireless channel model	Unit disk graph with distance loss
Communication range	80 m (Tx/Rx), 100 m (interface)
Node type	Tmote sky

illustrated in Figure 4. In this coupling, the Raspberry ensures both the RPL root for all dispatched instances and the bridging between the WSN and Internet. It also communicates with the management PC which does not only forward model parameters but also receives open/closed gateway orders from it.

To interface with the WSN side, the TelosB mote runs a basic SLIP application[19] that reads raw bits of information from the IEEE802.15.4 radio interface and forwards them to the device's USB port and vice versa.

Once started, the gateway disseminates configuration parameters of all existing *regular* instances using DIO messages. As a result, nodes belonging to these instances can choose their best parent according to the selected RPL OF.[14] MRHOF[16] with ETX metric as the OF is used in the proposed implementation.

It is worth pointing out that when a new instance is created, the gateway advertises this one with the dedicated instance, thanks to the multicast group. Any newly deployed node (not yet configured instance) can get one and starts normal operation (application running and vicinity information gathering/feedback).

Use case

To begin, the evaluation of the network formation and scaling were done through simulations. Given that the simulated prototype we have developed currently supports only one gateway, a network, as large as the host computer's performance allows, has been setup. So, support for multiple instances on the gateway and end-to-end communication between WSN nodes and Internet are also evaluated.

Besides, a real scenario was deployed to evaluate the operation of the proposed model on multiple gateways. The use case has been applied to home automation for gathering environmental data.

Large-scale simulation. Using COOJA,[20] the simulator integrated to Contiki, a $500\,m \times 500\,m$ random network topology of up to 100 nodes grouped in three instances is setup. Sensor nodes on the simulator emulate the same application firmware as the physical TelosB devices. The gateway program is run on the host computer which communicates with the simulator (through a particular node running SLIP application) using a standard TCP socket. The simulation computer has the following hardware features: Core 2 Extreme QX6850 processor board clocked at 3 GHz, 8 GB main memory, and running Ubuntu 16.04 LTS operating system. For simplicity, instance IDs are same as the number of nodes operating on it, namely, 50, 30, and 20. Table 2 summarizes the protocol stack and the main parameters of the simulation environment.

Figure 5 shows the described network, where nodes are colored yellow, purple, and blue according to their instance ID, respectively, 50, 30, and 20. The green node at the center of topology is the one that connects the simulator with the gateway on the host computer and interfaces the border router. Arrows depict the next hop selected by sensor nodes according to the RPL OF. As desired, nodes build three separate routing trees depending on their respective color.

Browsing through gateway IPv6 address on the host computer enables to display sensors of any group as well as the statistics of upward and downward streams for every sensor. For example, Figure 6 shows that almost all instances, 20 sensors, are discovered and can be *pinged* from any host connected to the local network.

Real-world scenario. A prototype of the proposed model has been deployed using home automation application scenario. By way of demonstration, two applications were developed. The first collects temperature data and the second the brightness. The first application is registered with RPL instance 10 and the second with 20. Nodes run Contiki v 2.7 operating system[21] and the gateway used the Raspbian 4.4 kernel version. The 6-LBR gateway application[22] has been modified and extended to handle several RPL instances, as well as open/closed gateways according to network conditions and management PC commands. Two gateway devices (each consisting of the association of Raspberry and TelosB) and 10 sensors are placed at various locations in a five-room house. Gateways are connected to the Internet through a wired Ethernet backbone as shown in Figure 4.

Model parameters are set so that both gateways can capture half of the sensor nodes each, that is, $\alpha = [5, 5]$. Moreover, it is required that nodes reach any gateway through at most two hops, that is, $MAX_{Deep} = 2$. It is worth to note that in the executed scenario with a noiseless context, nodes are within the communication range of each other. Indeed, when the ETX routing metric (i.e. transmission reliability) is used, the path through a neighbor may be more interesting than the direct link (less reliable).

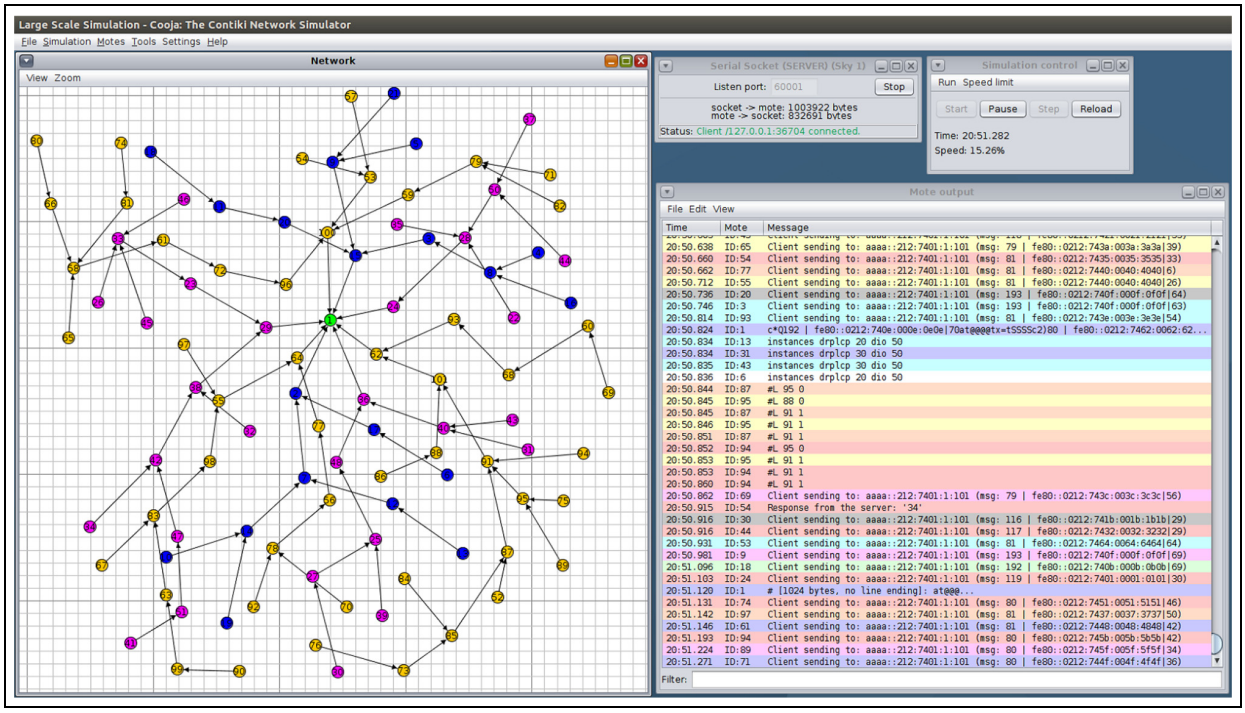
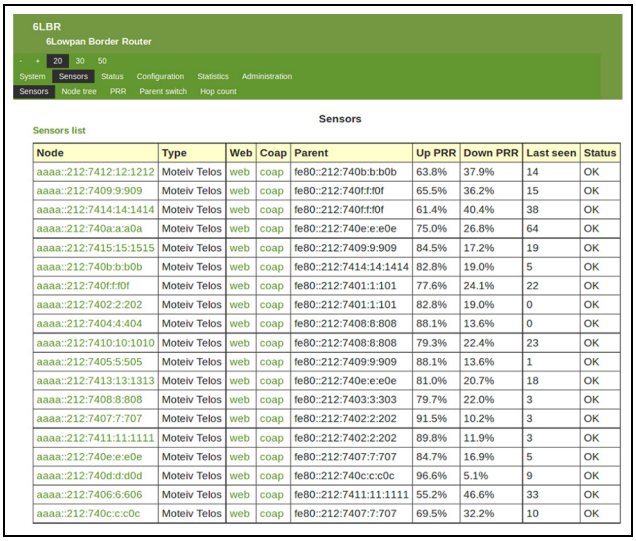


Figure 5. COOJA simulation of a large-scale scenario.



Node	Type	Web	Coap	Parent	Up PRR	Down PRR	Last seen	Status
aaaa::212:7412:12:1212	Moteiv Telos	web	coap	fe80::212:740b:b:b0b	63.8%	37.9%	14	OK
aaaa::212:7409:9:909	Moteiv Telos	web	coap	fe80::212:740f:f:f0f	65.5%	36.2%	15	OK
aaaa::212:7414:14:1414	Moteiv Telos	web	coap	fe80::212:740f:f:f0f	61.4%	40.4%	38	OK
aaaa::212:740a:a:a0a	Moteiv Telos	web	coap	fe80::212:740e:e:e0e	75.0%	26.8%	64	OK
aaaa::212:7415:15:1515	Moteiv Telos	web	coap	fe80::212:7409:9:909	84.5%	17.2%	19	OK
aaaa::212:740b:b:b0b	Moteiv Telos	web	coap	fe80::212:7414:14:1414	82.8%	19.0%	5	OK
aaaa::212:740f:f:f0f	Moteiv Telos	web	coap	fe80::212:7401:1:101	77.6%	24.1%	22	OK
aaaa::212:7402:2:202	Moteiv Telos	web	coap	fe80::212:7401:1:101	82.8%	19.0%	0	OK
aaaa::212:7404:4:404	Moteiv Telos	web	coap	fe80::212:7408:8:808	88.1%	13.6%	0	OK
aaaa::212:7410:10:1010	Moteiv Telos	web	coap	fe80::212:7408:8:808	79.3%	22.4%	23	OK
aaaa::212:7405:5:505	Moteiv Telos	web	coap	fe80::212:7409:9:909	88.1%	13.6%	1	OK
aaaa::212:7413:13:1313	Moteiv Telos	web	coap	fe80::212:740e:e:e0e	81.0%	20.7%	18	OK
aaaa::212:7408:8:808	Moteiv Telos	web	coap	fe80::212:7403:3:303	79.7%	22.0%	3	OK
aaaa::212:7407:7:707	Moteiv Telos	web	coap	fe80::212:7402:2:202	91.5%	10.2%	3	OK
aaaa::212:7411:11:1111	Moteiv Telos	web	coap	fe80::212:7402:2:202	89.8%	11.9%	3	OK
aaaa::212:740e:e:e0e	Moteiv Telos	web	coap	fe80::212:7407:7:707	84.7%	16.9%	5	OK
aaaa::212:740d:d:d0d	Moteiv Telos	web	coap	fe80::212:740c:c:c0c	96.6%	5.1%	9	OK
aaaa::212:7406:6:606	Moteiv Telos	web	coap	fe80::212:7411:11:1111	55.2%	46.6%	33	OK
aaaa::212:740c:c:c0c	Moteiv Telos	web	coap	fe80::212:7407:7:707	69.5%	32.2%	10	OK

Figure 6. Discovered sensor nodes at gateway (instance 20).

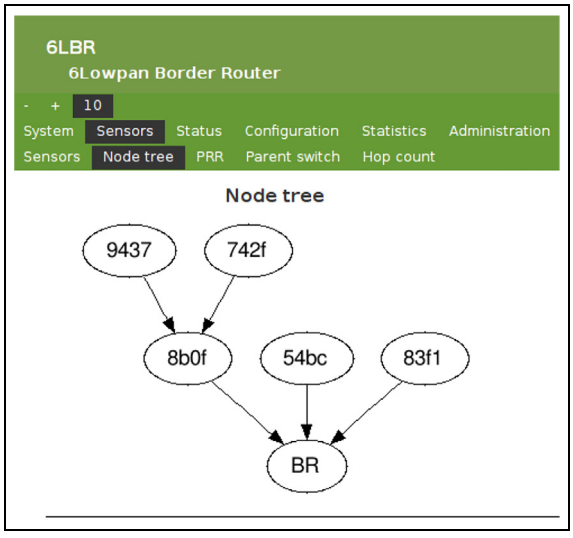


Figure 7. Home automation scenario: first gateway—stage 1.

First of all, five sensor nodes running instance 10 related to the first application were deployed. Instances were previously created at gateway level. After the initialization step and given the above parameters, the proposed model advocates the opening of only one device, denoted as gateway 1. Figure 7 shows the network topology graph obtained from web interface used to interact with the first Raspberry Pi. As shown, all five sensor nodes materialized by their 16-bit MAC-ID use this device as the only exit point to Internet. Indeed, the corresponding outputs computed by the proposed model are $X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ and $Y = [1, 0]$.

As a next step, the second instance 20 was created on gateways and the remaining five nodes were deployed, so that two execute the first instance application and others, the second. Then, gateway 2 (previously closed) starts to act as RPL root for both instances. As shown,

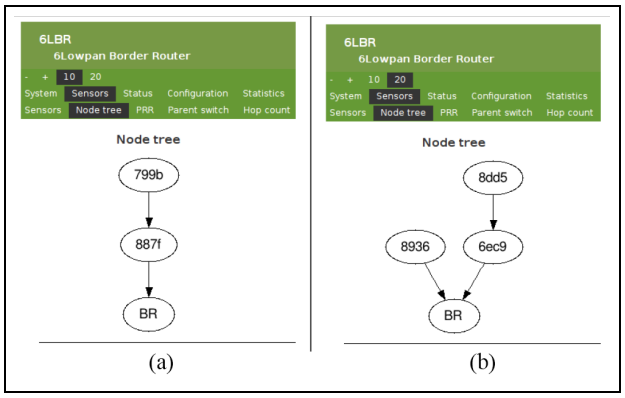


Figure 8. Home automation scenario: second gateway—stage 2: (a) instance 10 app and (b) instance 20 app.

when instance tabs are browsed, two nodes operate for 10 (Figure 8(a)) and three for 20 (Figure 8(b)). Note that the first gateway continues to capture the previously deployed nodes as highlighted above (Figure 7).

Conclusion

This article investigates how to select gateways for efficient integration of WSNs to Internet with the aim of achieving the IoT vision. A backbone architecture that connects several multi-instance applications has also been designed. The central point on the targeted framework is the *gateway*. An architecture has been designed to be flexible and it involves low hardware costs. At the same time, its functionality must be guaranteed. ILP was used to determine which gateways (i.e. number and position) are suitable for enabling connection between WSN and the Internet, so that application bandwidth and gateway capacity are met while avoiding network bottlenecks.

A prototype of the proposed model was implemented as proof of a concept using low-cost Raspberry Pi as the gateway. A real-world application scenario for home automation was deployed to demonstrate the effectiveness of the proposed scheme. The COOJA simulator also showed that the architecture can scale up to 100 sensors managed by the same gateway. We argue that a possible limitation (i.e. simulation size) can only relate to memory resources, since the sensor firmwares are fully emulated.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Patrick Olivier Kamgueu https://orcid.org/0000-0002-2669-2872

References

1. Mattern F and Flörkemeier C. Vom internet der computer zum internet der dinge. *Inform Spektrum* 2010; 33(2): 107–121.
2. Zachariah T, Klugman N, Campbell B, et al. The Internet of Things has a gateway problem. In: *Proceedings of the 16th international workshop on mobile computing systems and applications*, Santa Fe, NM, 12–13 February 2015, pp.27–32. New York: ACM.
3. Wang L, Teng H and Yu GH. Sensors access scheme design based on internet of things gateways. In: *Proceedings of the fifth international conference on intelligent systems design and engineering applications*, Hunan, China, 15–16 June 2014, pp.901–904. New York: IEEE.
4. Bimschas D, Hellbrück H, Mietz R, et al. Middleware for smart gateways connecting sensornets to the internet. In: *Proceedings of the fifth international workshop on middleware for sensor networks*, Bangalore, India, 30 November 2010, pp.8–14. New York: ACM.
5. Waharte S, Boutaba R and Anelli P. Impact of gateways placement on clustering algorithms in wireless mesh networks. In: *Proceedings of IEEE international conference on communications*, Dresden, 14–18 June 2009, pp.1–5. New York: IEEE.
6. Wu W, Luo J and Yang M. Gateway placement optimization for load balancing in wireless mesh networks. In: *Proceedings of the 13th international conference on computers supported cooperative work in design*, Santiago, Chile, 22–24 April 2009, pp.408–413. New York: IEEE.
7. Luo J, Wu W and Yang M. Optimization of gateway deployment with load balancing and interference minimization in wireless mesh networks. *J Univ Comput Sci* 2011; 17(14): 2064–2083.
8. Boubrima A, Bechkit W and Rivano H. Optimal WSN deployment models for air pollution monitoring. *IEEE T Wirel Commun* 2017; 16(5): 2723–2735.
9. Petrolo R, Morabito R, Loscrì V, et al. The design of the gateway for the cloud of things. *Annales Telecommun* 2017; 72(1–2): 31–40.
10. Raspberry Pi Foundation, http://www.raspberrypi.org (accessed 15 June 2016).
11. Shetty B. Approximate solutions to large scale capacitated facility location problems. *Appl Math Comput* 1990; 39(2): 159–175.
12. Zhang J, Chen B and Ye Y. A multiexchange local search algorithm for the capacitated facility location problem. *Math Oper Res* 2005; 30(2): 389–403.
13. IBM ILOG CPLEX Optimizer 12.6.3, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (accessed 10 July 2016).
14. Winter T, Thubert P, Brandt A, et al. Rpl: Ipv6 routing protocol for low-power and lossy networks. RFC 6550 (Proposed Standard), March 2012, https://www.rfc-editor.org/rfc/rfc6550.txt

15. Thubert P. Objective function zero for the routing protocol for low-power and lossy networks (RPL). RFC 6552 (Proposed Standard), March 2012, https://www.rfc-editor.org/rfc/rfc6552.txt
16. Gnawaliand O and Levis P. The minimum rank with hysteresis objective function. RFC 6719 (Proposed Standard), September 2012, https://www.rfc-editor.org/rfc/rfc6719.txt
17. Kamgueu PO, Nataf E, Djotio TN, et al. Energy-based metric for the routing protocol in low-power and lossy network. In: *Proceedings of the 2nd international conference on sensor networks*, Barcelona, Spain, February 2013, pp.145–148. SciTePress.
18. Kamgueu PO, Nataf E and Djotio TN. On design and deployment of fuzzy-based metric for routing in low-power and lossy networks. In: *Proceedings of the 40th IEEE local computer networks conference workshops*, Clearwater Beach, FL, 28–29 October 2015, pp.789–795. New York: IEEE.
19. Romkey JL. Nonstandard for transmission of IP datagrams over serial lines: Slip. RFC 1055 (Internet Standard), June 1988, https://www.rfc-editor.org/rfc/rfc1055.txt
20. Österlind F, Dunkels A, Eriksson J, et al. Cross-level sensor network simulation with COOJA. In: *Proceedings of the 31st annual IEEE conference on local computer networks*, Tampa, FL, 14–16 November 2006, pp.641–648. New York: IEEE.
21. Dunkels A, Grönvall B and Voigt T. Contiki—a lightweight and flexible operating system for tiny networked sensors. In: *Proceedings of the 29th annual IEEE conference on local computer networks*, Tampa, FL, 16–18 November 2004, pp.455–462. New York: IEEE.
22. Deru L, Dawans S, Ocaña M, et al. Redundant border routers for mission-critical 6LoWPAN networks. In: *Proceedings of the fifth workshop on real-world wireless sensor networks*, Como Lake, 2013, pp.195–203, https://inl.info.ucl.ac.be/system/files/deru13redundant.pdf